C++ pour la robotique

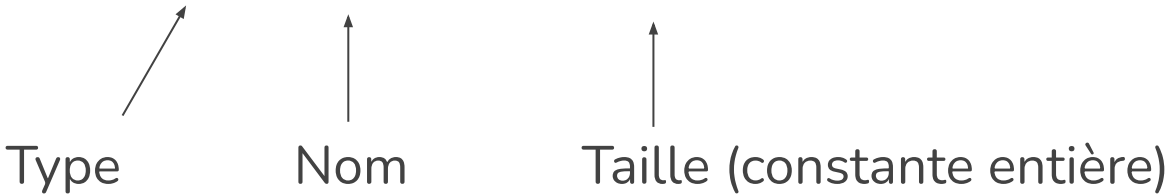
C++ - Tableaux

Sébastien Rothhut

Les tableaux

Structure de données regroupant des éléments du même type

Déclaration : `int position_x[50];`



Les tableaux

Accès aux éléments du tableau :

`position_x[12]` ← Référence à l'élément d'indice 12

Les indices vont de 0 à *<Taille du tableau>* - 1

Les tableaux

Déclaration - initialisation :

```
int tableau[3] = {2, 4, 6};
```

Initialisation après déclaration :

```
const int taille_tableau = 3;
int tableau[taille_tableau];
for (int i = 0; i < taille_tableau; i++) {
    tableau[i] = (i+1)*2;
}
```

Les tableaux

Déclaration - initialisation sans spécifier la taille :

```
int tableau[3] = {2, 4, 6}; // 3 éléments

int tableau[] = {2, 4, 6}; // 3 éléments

int tableau[5] = {2, 4, 6}; // 5 éléments, 3 initialisés
```

Les tableaux

Affectation :

```
tableau1 = tableau2;
```

Passage en paramètre d'une fonction : accès

```
void afficher(int tableau[], int taille_tableau) {
    for(int i = 0; i < taille_tableau; i++) {
        cout << tableau[i] << endl;
    }
}
```

Les tableaux

Passage en paramètre d'une fonction : modification*

```
void incrementer(int tableau[], int taille_tableau) {
    for(int i = 0; i < taille_tableau; i++) {
        tableau[i] = tableau[i] + 1;
    }
}

int t[3] = {1, 2, 3};
incrementer(t, 3);
cout << t[2] << endl;
```

Les tableaux

Valeur de retour d'une fonction:

```
int[] f() {
    int tab[3] = {1, 2, 3};
    return tab;
}
```

Exercice

https://www.onlinegdb.com/online_c++_compiler/

On veut représenter les notes d’un étudiant au cours de l’année

Ecrire un programme qui :

- Déclare un tableau de 10 nombres représentant les notes d’un étudiant
- Remplit le tableau interactivement
- Calcule la moyenne des notes
- Affiche la moyenne

Exercice

On veut représenter les notes d’un étudiant au cours de l’année

Ecrire une classe basée sur le tableau de notes précédent, qui permet de gérer une liste avec les fonctions suivantes :

- Ajout d’une note à la suite de la précédente (avec gestion des bornes)
- Suppression de la dernière note (avec gestion des bornes)
- Récupérer la valeur de la nème note (n compris entre 0 et 9) ou -1 si erreur
- Calcul de la moyenne des notes ajoutées à la liste

Chaînes de caractères

Déclaration - initialisation :

```
char mot[3] = {‘m’, ‘o’, ‘t’};
```

~~`char mot[3] = “mot”;`~~

```
main.cpp: In function ‘int main()’:
main.cpp:38:19: error: initializer-string for array of
chars is too long [-fpermissive]
   38 |     char mot[3] = "mot";
      |                   ^~~~~
```

Chaînes de caractères

Déclaration - initialisation :

```
char mot[4] = “mot”;
```

```
char mot[4] = {‘m’, ‘o’, ‘t’, ‘\0’};
```

Pour pouvoir être manipulées facilement par des fonctions, les chaînes de caractères sont supposées terminées par un caractère ASCII de code 0 (0 sous forme d’entier, ou ‘\0’ sous forme de caractère).

Chaînes de caractères

Fonctions utilitaires :

```
#include <cstring>
char nomCarte[30];
strcpy(nomCarte, "Arduino"); // string copy
strcat(nomCarte, “ Uno”); // concatenation
cout<<strlen(nomCarte)<<endl; // string length
```

Exercice

On veut représenter des étudiants d'une promo.

- Ecrire une classe Etudiant avec comme attributs nom et prenom, de type char[] (d'une taille arbitraire)
- Définir un constructeur qui prend en paramètres deux char[] afin d'initialiser le nom et le prénom
- Définir une fonction membre retournerNom qui permet de récupérer le nom complet de l'étudiant dans un char[]

Matrices

Tableaux de dimension > 1 :

- Déclaration :

```
int matrice[2][3]; // 2 lignes, 3 colonnes
```

- Accès :

```
matrice[i][j];
```

Matrices

- **Déclaration - initialisation** :

```
int matrice[2][3] = {{1, 3, 5}, {1, 2, 4}};

char phrase[7][15] = {"Goodbye", "and",
"thanks", "for", "all", "the", "fish"};

char phrase[][15] = {"Goodbye", "and",
"thanks", "for", "all", "the", "fish"};
```

Exercice

On veut lister les étudiants d’une promo.

- Déclarer une chaîne de caractères qui représente un panneau d’affichage, d’une capacité de 2000 caractères
- Déclarer un tableau d’objets Etudiant et le remplir de quelques instances
- Définir une fonction qui écrit dans le panneau les noms des étudiants, séparés par un saut de ligne (‘\n’)

Pointeurs

Adressage en mémoire :

```
int var = 12;
```

Adresse	0	1	2	...	66288	66289	66290
Mémoire					12		

```
cout << &var;
```

Pointeurs

Pointeur = variable typée contenant l'adresse d'une autre variable

```
int *p = &var;
```

Adresse	0	1	2	...	**66288**	66289	66290	...	87612
Mémoire					12				**66288**

Pointeurs

Déclaration :

```
int *p; // valeur indéterminée
```

```
int *p = nullptr; // C++11
```

Initialisation : `p = &var;`

Pointeurs

Accès à la valeur pointée : `cout << *p << endl;`

Vérification :

```
if (p) {
    cout << “le pointeur est initialisé”;
}
```

Modification de la valeur pointée : `*p = 2;`

Pointeurs

Modification du pointeur :

```
int *p1, *p2;
p2 = p1;
```

Attention au typage :

```
char *p_char;
int *p_int;
p_int = p_char; // ne compile pas
```

Tableaux et pointeurs

Stockage en mémoire :

```
char tableau[4] = “abc”;
```

Adresse	…	66288	66289	66290	66291	…
Mémoire		‘a’	‘b’	‘c’	‘\0’	

Tableaux et pointeurs

Stockage en mémoire :

```
uint16_t tableau[4] = {1, 2, 3, 4};
```

Adresse	...	66288	66290	66292	66294	...
Mémoire		1	2	3	4	

Tableaux et pointeurs

Equivalence tableaux - pointeurs :

```
uint16_t tableau[4] = {1, 2, 3, 4};
```

`tableau` :

- est un pointeur constant sur un `uint16_t`
- contient l'adresse du premier élément du tableau

Tableaux et pointeurs

Equivalence tableaux - pointeurs :

```
uint16_t tableau[4] = {1, 2, 3, 4};
uint16_t *p = tableau;
for(int i = 0; i < 4; i++) {
    cout << p[i] << endl;
}
```

Tableaux et pointeurs

```
void incrementer(int tableau[], int taille_tableau) {
    for(int i = 0; i < taille_tableau; i++) {
        tableau[i] = tableau[i] + 1;
    }
}

int t[3] = {1, 2, 3};
incrementer(t);
cout << t[3] << endl;
```

Exercice

Écrire une fonction qui a comme paramètres

- un tableau d'entiers de taille quelconque,
- la taille du tableau,
- 2 pointeurs vers des entiers min et max.

La fonction doit renvoyer dans les entiers pointés par min et max respectivement le plus petit et le plus grand entier du tableau.

Exercice

```
int minmax(int *p, int n, int *pmin, int *pmax)
{
    int *i(0);
    *pmin=*p;
    *pmax=*p;

    for (i=p ; i<p+n ; i++)
    {
        if (*i < *pmin) *pmin = *i;
        if (*i > *pmax) *pmax = *i;
    }
    return 0;
}
```

Exercice

```
int main()
{
    int tn[] = { 12, 23, 36, 5, 46, 9, 25 };
    int min, max, r;

    r = minmax(tn, sizeof(tn)/sizeof(int), &min, &max);
    cout << "Min, Max : " << min << " " << max << endl;
    return 0;
}
```

Exercice

Écrire une fonction qui a en paramètre une chaîne de caractères et qui renvoie par un return le nombre d'occurrences de la lettre 'A'. Cette fonction devra parcourir la chaîne en utilisant un pointeur.

Tester cette fonction.

Exercice

```c
int nba(char *p)
{
    int n=0;
    while (*p != '\0')
    {
        if (*p =='A') n++;
        p++;
    }
    return n;
}
```

Exercice

```
int main()
{
    char ss[50];
    cout << "Saisissez une chaine de caractères :
" << endl;
    cin.getline(ss, 50);
    cout << "Vous avez saisi : " << ss << endl;
    cout << "La chaine comporte " << nba(ss) << "
A." << endl;
    return 0;
}
```

Variables dynamiques

Déclaration :

```
int *p; // valeur indéterminée
```

p ne pointe sur rien, aucun espace mémoire n'a été réservé

```
p = new int;
```

p a une adresse, l'espace mémoire peut contenir un int

Variables dynamiques

Initialisation :

```
*p = 42;
```

p pointe sur une **variable dynamique**. L'espace mémoire reste réservé même après être sorti du bloc dans lequel on a déclaré p.

Variables dynamiques

Désallocation :

```
delete p;
```

l'espace mémoire est libéré, on ne peut plus accéder au contenu pointé *p

Tableaux dynamiques

```
int *tableau; // aucune mémoire allouée

tableau = new int[n]; // n peut être
déterminé au moment de l'exécution

delete[] tableau; // désallocation de la
mémoire allouée au tableau
```